Monte Carlo Response-Time Analysis

Sergey Bozhko, Georg von der Brüggen, and Björn B. Brandenburg
Max Planck Institute for Software Systems (MPI-SWS)

***Abstract*—Determining a *soft* or *firm* real-time task's *probabilistic worst-case response time* is a central goal when quantifying and bounding the probability of deadline misses, but current approaches are either (i) fast, but coarse-grained analytical bounds without precision guarantees, (ii) based on convolution and suffer from high space and time complexity, or (iii) combine convolution with resampling techniques that accrue pessimism in an uncontrolled manner. As a new alternative, this paper provides the first probabilistic response-time analysis method based on Monte Carlo simulation, which provides a controlled trade-off between analysis runtime, the desired degree of accuracy, and the permissible probability of a misestimate. An evaluation shows the proposed Monte Carlo analysis to routinely provide more accurate worst-case deadline failure probability (WCDFP) estimates than prior approaches, especially when considering large task sets (where prior methods struggle). In particular, it is shown to scale to workloads with up to 500 tasks while achieving one to three orders of magnitude better precision than analytical or convolution-based approaches (given an equivalent time budget).**

I. Introduction

In most "every-day" real-time systems, occasional deadline misses do not cause disaster or complete failure of the system. In the real-time systems literature, such systems are classified as *soft* or *firm*. According to a recent survey of industry practice in real-time systems [3], about two thirds (62%) of the respondents indicated that the system they are working on includes soft or firm timing constraints, and only 5% indicated that they work on a system with purely hard timing constraints. Another interesting finding [3] is that, when asked about the maximum allowed frequency at which the *most time-critical* function of a system can miss its deadlines, 45% of the respondents indicated that said functionality can miss some deadlines (ranging from 1 in 10 to 1 in 1 billion), 35% were unable to give a specific answer, and only 15% stated that deadlines cannot be missed.

Nonetheless, even in soft real-time systems, too-frequent deadline misses can cause unacceptable performance degradation; their likelihood should hence be assessed *a priori*. One of the central metrics in this context is the *worst-case deadline failure probability* (WCDFP) [17] (defined formally in Section VI). Intuitively, this notion corresponds to an upper bound on the probability of observing the *first* deadline miss of a task under analysis in a given busy window. The WCDFP allows bounding the average time to (temporal) failure of a system and, if jobs that miss their deadlines are immediately aborted by the system, directly corresponds to an upper bound on the expected deadline-miss rate. Furthermore, even if tardy jobs are not aborted, the WCDFP still plays an important role when estimating the deadline-miss rate [11].

A common way to determine the WCDFP is to perform a probabilistic response-time analysis [39, 41, 57]. In contrast to classic response-time analyses based on a scalar *worst-case execution time* (WCET) [4, 7, 18, 27, 32], which conservatively assume each task activation (or job) to exhibit the worst possible execution time, probabilistic response-time analyses consider a distribution of potential execution times for each task and, based on this, estimate the resulting *response-time distribution*. Such response-time distributions are then used to derive the probability that a task misses a deadline.

Probabilistic response-time analyses are a basic building block for many probabilistic techniques and, in addition to bounding the WCDFP, are used in numerous other related problems such as analyses of controller area networks [5, 49], preemption-point selection [39], correlated response times [54], and fault tolerance [8, 16, 48].

Unfortunately, a fundamental complication in any probabilistic response-time analysis is that a probabilistic response time is ultimately a sum of (many) random variables. Thus, a direct, *exact* computation via naïve convolution would result in extreme growth of the representation of the resulting distribution and is practically infeasible. Consequently, prior analyses have been driven towards faster techniques that safely upper-bound the response-time distribution [10, 12, 42, 43, 50, 57]. These methods can be informally divided into two families of approaches: convolution-based approaches and analytical bounds.

The *convolution-based approaches* trace their origin back to Díaz et al.'s concept of *intentional pessimism* in the probabilistic analysis of real-time systems [20]. By defining a partial order on random variables that reflects approximation safety, Díaz et al. enabled a lossy, but sound and more scalable over-approximation of the resulting response-time distribution, which has since been reused in several papers [42, 43, 50]. In particular, this idea was further developed by Refaat and Hladik [50] and Maxim and Cucu-Grosjean [41], who proposed to *resample* after each convolution step such that the resulting distribution is upper-bounded by a distribution with a more compact representation. This allows controlling the size of the final distribution at the cost of some pessimism. However, the pessimism of the resulting distribution depends in non-obvious ways on the choice of resampling procedure and the number of points in the representation of the resampled distribution. Moreover, the pessimism tends to accumulate and compound. To date, no bound on the final pessimism has been proposed and there is no known way of ensuring that the pessimism remains below a given threshold.

Analytical upper bounds circumvent the need to actually obtain a response-time distribution and instead directly bound the proportion of the distribution that exceeds the relative deadline. These approaches make use of analytical upper

bounds on random variables such as Chernoff's [13] (Chen et al. [10, 12]), Bernstein's [6], or Hoeffding's [26] inequalities (von der Brüggen et al. [57]). These inequalities are general theorems about the probability of observing a given deviation of the sum of multiple random variables from its expected value that do not involve computing the underlying distribution. However, while yielding an upper bound quickly, these methods are also known to be quite pessimistic in many cases [12, 57].

In this paper, we aim for a middle ground between the two families of prior approaches. To this end, we explore the application of *Monte Carlo methods* (*i.e.*, probabilistic algorithms). The distinguishing feature of such algorithms is that they may output incorrect results with (arbitrarily) low probability for the benefit of significantly improved runtimes compared to deterministic algorithms.

More precisely, we present a novel approach based on Monte Carlo simulation that, at the cost of a configurable and bounded probability of a misestimate, provides two desirable features: **(i)** it allows estimating the probabilistic response-time of a job significantly faster than known convolution-based approaches, and **(ii)** in contrast to analytical bounds, it allows fine-grained control over the achieved precision.

Contributions. We consider the problem of determining the WCDFP of a task under uniprocessor fixed-priority preemptive scheduling when each task's execution time is defined using a discrete distribution. Our main contributions are:

- We provide a novel approach based on Monte Carlo simulation (Sections IV and V) that allows estimating the probabilistic response-time of a given job with controllable precision and controllable probability of a misestimate.
- We show how the proposed Monte Carlo method can be used to bound the WCDFP under fixed-priority preemptive uniprocessor scheduling (Section VI).
- We conducted an evaluation with randomized task sets (Section VII), demonstrating scalability to large sets with up to 500 tasks and improved accuracy compared to the state of the art: on average, the proposed approach was 1–3 orders of magnitude more precise than analytical or convolution-based methods (given the same time budget).

II. System Model and Background

We begin by describing the general system model and recalling some basic definitions on which our analysis rests. We assume a discrete-time model, where the smallest quantity $\gamma > 0$ represents an indivisible unit of time (*e.g.*, a processor cycle) and $\mathbb{T} \triangleq \{\gamma \cdot k \mid k \in \mathbb{N}\} \subset \mathbb{R}$ denotes the time domain.

A. Probability Theory Primer

To start, we briefly review key concepts of probability theory.

Def. 1. A *probability space* is a triple $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω is the set of all possible *outcomes* of an experiment, $\mathcal{F}$ is a set of *events*, where each event is a subset of Ω, and $\mathbb{P} : \mathcal{F} \to [0, 1]$ is a *probability function*. If not specified otherwise, we assume that Ω is discrete and $\mathcal{F}$ is the powerset of Ω (*i.e.*, $\mathcal{F} = 2^{\Omega}$).

In our context, a random variable is a function that maps elements of the set of possible outcomes to a discrete set.

Def. 2. A *random variable* is a function $\mathcal{X} : \Omega \to E$ from a set of possible outcomes to a space E. In this paper, E is a discrete subset of $\mathbb{R}$ or $\mathbb{R}^k$ such as $\{0, 1\}$, $\mathbb{N}$, $\mathbb{T}$, or $\mathbb{T}^k$.

For notational brevity, it is common to omit the argument of a random variable and use the expression $\mathbb{P}[\mathcal{X} = x]$ as an abbreviation for $\mathbb{P}[\{\omega \in \Omega | \mathcal{X}(\omega) = x\}]$. Similarly, the expression $\mathbb{P}[\mathcal{X} \leq x]$ is an abbreviation for $\mathbb{P}[\{\omega \in \Omega | \mathcal{X}(\omega) \leq x\}]$. Furthermore, two random variables $\mathcal{X}$, $\mathcal{Y}$ are *independent* if $\mathbb{P}[\mathcal{X} = x \wedge \mathcal{Y} = y] = \mathbb{P}[\mathcal{X} = x] \cdot \mathbb{P}[\mathcal{Y} = y]$.

Next, we define the cumulative distribution function.

Def. 3. Given a random variable $\mathcal{X} : \Omega \to E$, where $E \subseteq \mathbb{R}$, its *cumulative distribution function* (CDF) $F[\mathcal{X}] : \mathbb{R} \to [0, 1]$ is defined as $F[\mathcal{X}](x) \triangleq \mathbb{P}[\mathcal{X} \leq x]$.

There is a simple relation between the equality of probabilities and the equality of CDFs of two discrete random variables.

Fact 4. *For any two random variables $\mathcal{X}$ and $\mathcal{Y}$ with codomain $\mathbb{T}$, $F[\mathcal{X}] = F[\mathcal{Y}]$ if and only if $\mathbb{P}[\mathcal{X} = t] = \mathbb{P}[\mathcal{Y} = t]$ for $t \in \mathbb{T}$.*

We adopt the notion of a partial order on random variables from Díaz et al. [20]. In the following sections, the terms "upper bound" or "maximum" in the context of random variables or distributions should always be understood in the sense of Def. 5.

Def. 5 (from [20])**.** Let $\mathcal{X}_1$ and $\mathcal{X}_2$ be two random variables. We say that $\mathcal{X}_1$ is *greater than or equal to* $\mathcal{X}_2$ if $F[\mathcal{X}_1](x) \leq F[\mathcal{X}_2](x)$ for any $x \in \mathbb{R}$, and denote this relation by $\mathcal{X}_1 \succeq \mathcal{X}_2$.

Next, we define the inverse cumulative distribution function.

Def. 6. Given a random variable $\mathcal{X} : \Omega \to E$, where $E \subseteq \mathbb{R}$, its *inverse cumulative distribution function* (ICDF) $Q[\mathcal{X}] : [0, 1] \to \mathbb{R}$ is defined as $Q[\mathcal{X}](q) \triangleq \inf\{x \in \mathbb{R} \mid F[\mathcal{X}](x) \geq q\}$.

The first key result that plays an important role later in the paper is the *inverse transform sampling method*, which allows sampling from a complex distribution if the ICDF is known. We use the following notation: given a random variable $\mathcal{X} : \Omega \to E_1$ and a function $f : E_1 \to E_2$, we let $f(\mathcal{X}) : \Omega \to E_2$ denote a random variable that maps each $\omega \in \Omega$ to $f(\mathcal{X}(\omega))$.

Theorem 7 (Lemma 2.4 in [51])**.** *Let $\mathcal{X}$ be a random variable with cumulative distribution function $F[\mathcal{X}]$. If a random variable $\mathcal{Y}$ has a uniform distribution $\mathcal{U}([0, 1])$, then the random variable $Q[\mathcal{X}](\mathcal{Y})$ has the same distribution as $\mathcal{X}$.*

Second, we recall that a function of multiple random variables can be substituted with an equivalent expression involving only individual random variables. To express this concisely, let $[\![x]\!]$ denote the *indicator function* that evaluates to 1 if x is true, and to 0 otherwise. Furthermore, given a function f with domain $\mathbb{T}^k$, let $\sum_{x_1,\ldots,x_k} f(x_1, \ldots, x_k) \triangleq \sum_{x_1 \in \mathbb{T}} \sum_{x_2 \in \mathbb{T}} \cdots \sum_{x_k \in \mathbb{T}} f(x_1, \ldots, x_k)$ denote the sum over all possible tuples $(x_1, \ldots, x_k)$ in $\mathbb{T}^k$.

Fact 8. *For a function $f(x_1, \ldots, x_k) : \mathbb{T}^k \to \mathbb{T}$ and k independent random variables $\mathcal{X}_1, \ldots, \mathcal{X}_k$, $\mathbb{P}\left[f(\mathcal{X}_1, \ldots, \mathcal{X}_k) = x\right] = \sum_{x_1,\ldots,x_k} [\![f(x_1, \ldots, x_k) = x]\!] \cdot \prod_{i=1}^{k} \mathbb{P}\left[\mathcal{X}_i = x_i\right]$.*

For the third and final major building block, recall that a random variable $\mathcal{X}$ is called a *Bernoulli trial* if it has exactly two outcomes that can be interpreted as "success" and "failure" and in which the probability of a success p is the same every time the experiment is conducted. Given a Bernoulli trial with probability of success p, the *binomial random variable* $\mathcal{B}(s, p)$ describes the probability of observing a given number of successes in s independent Bernoulli trials.

In practice, the success probability p is often unknown. To estimate the ground-truth success probability p from repeated observations of an experiment, *binomial proportion confidence intervals* are used. An interval $[l, r]$ is called a $(1-\varepsilon)$-confidence interval for parameter p if the probability that $p \in [l, r]$ is at least $1-\varepsilon$. One commonly used confidence interval is the Agresti–Coull interval [2], which we adopt, too.

Theorem 9 ([2])**.** *Given k successes in s trials, let $\tilde{s} \triangleq s + z^2$, and $\tilde{p} \triangleq \frac{1}{\tilde{s}}\left(k + \frac{z^2}{2}\right)$. Then a $(1-\varepsilon)$-confidence interval for p is given by $\tilde{p} \pm z\sqrt{\tilde{p}(1-\tilde{p})/\tilde{s}}$, where z is the $(1-\varepsilon/2)$-th quantile of the standard normal distribution (with mean 0 and variance 1), that is, $z \triangleq \Phi^{-1}\left(1 - \frac{\varepsilon}{2}\right)$.*

B. System Model

We consider a system comprised of n sporadic firm real-time tasks $\tau \triangleq \{\tau_1, \ldots, \tau_n\}$. We assume that a job that misses its deadline is immediately discarded (*i.e.*, removed from the system); we later revisit this assumption in Section VI. Each task $\tau_i \triangleq (F_i, T_i, D_i, \pi_i)$ is characterized by an *execution-time distribution* F_i, its *minimum inter-arrival time* T_i, its *relative deadline* D_i, and its *priority* π_i. We write $\pi_h \succeq \pi_l$ to denote that task τ_h has a priority equal to or exceeding that of task τ_l. We consider constrained deadlines: $D_i \leq T_i$ for any $\tau_i \in \tau$.

Whenever a task is activated, a corresponding job is released. Let $J_{i,j}$ denote the j-th job of task τ_i. Each job $J_{i,j}$ has a *release (or activation)* time $a_{i,j}$, an *absolute deadline* $d_{i,j} = a_{i,j} + D_i$, a *priority* equal to its task's priority π_i, and an *execution time* (or *cost*) $c_{i,j}$, where the possible values of $c_{i,j}$ are distributed in accordance with the job's (usually unknown) *probabilistic execution time* (pET) $\mathcal{C}^\star_{i,j}$, which describes the probability that the job exhibits a particular cost. We use the terms "execution time" and "job cost" interchangeably.

The task set is scheduled on a uniprocessor according to a fully preemptive fixed-priority scheduling policy. That is, at each point in time, the scheduler ensures that (one of) the ready job(s) with the highest priority is executed.

Unfortunately, pETs—which describe the probability that a job's *actual* execution time is equal to a given value—are exceedingly difficult to obtain and not necessarily independent random variables, which renders many analysis techniques inapplicable. To mitigate these challenges, it is common [15] to instead consider a probabilistic *worst-case* execution time that upper-bounds the actual pET in the sense of Def. 5.

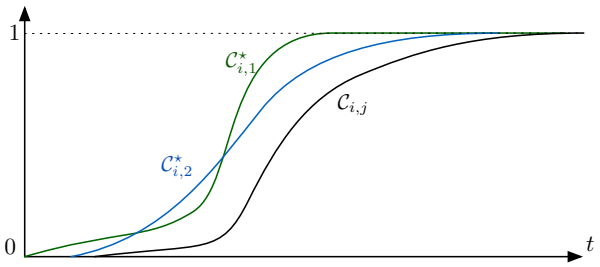


Fig. 1: The pWCET $\mathcal{C}_{i,j}$ upper-bounds the pETs $\mathcal{C}^\star_{i,1}$ and $\mathcal{C}^\star_{i,2}$.

Def. 10. The *probabilistic worst-case execution time* (pWCET) $\mathcal{C}_{i,j}$ of a job $J_{i,j}$ is a random variable with codomain $\mathbb{T}$ such that **(i)** it upper-bounds the pET of the job (*i.e.*, $\mathcal{C}_{i,j} \succeq \mathcal{C}^\star_{i,j}$), **(ii)** its distribution is equal to task τ_i's execution-time distribution (*i.e.*, $F[\mathcal{C}_{i,j}] = F_i$), and **(iii)** pWCETs corresponding to different jobs are independent random variables.

Since we require $F[\mathcal{C}_{i,j}] = F_i$, we can state Def. 10 in a different manner. Given an arbitrary constant $x \in \mathbb{T}$ and a job $J_{i,j}$ of task τ_i, it holds that $\mathbb{P}\left[\mathcal{C}^\star_{i,j} \leq x\right] \geq F_i(x)$. Visually, this means that the CDF of $\mathcal{C}_{i,j}$ stays below the CDF of $\mathcal{C}^\star_{i,j}$ for any j (see Fig. 1). We further assume that an ICDF Q_i corresponding to F_i is known for each τ_i. That is, there is a function $Q_i : [0, 1] \to \mathbb{R}$ such that for any quantile $q \in [0, 1]$ and any job $J_{i,j} \in \tau_i$: $\mathbb{P}\left[\mathcal{C}^\star_{i,j} \leq Q_i(q)\right] \geq q$.

For notational clarity, we introduce the *arrival sequence* $\xi(t) \triangleq \{J_{i,j} \mid J_{i,j} : a_{i,j} = t\}$, which maps each instant t to the set of jobs that arrive at t. Based on ξ, we introduce three auxiliary notions: the *arrival sequence of a task* τ_i, defined as $\xi_i(t) \triangleq \xi(t) \cap \{J_{i,j} \mid j \in \mathbb{N}\}$, the *higher-or-equal priority arrival sequence of* τ_i, defined as $\xi_{\succeq i}(t) \triangleq \bigcup_{h:\pi_h \succeq \pi_i} \xi_h(t)$, and their extensions to intervals, *e.g.*, $\xi([t_1, t_2)) \triangleq \bigcup_{t \in [t_1, t_2)} \xi(t)$.

III. Formalization of the Response-Time Bound

To lay a solid foundation for the subsequent development of our method and its proof of correctness, we take a short detour and derive the notion of a "probabilistic response-time bound" from first principles. What we present is a simple construction of a natural upper bound on the "probabilistic response time" of a job. While arguably similar definitions were used *implicitly* in many earlier papers on this topic [15, 19, 34, 39, 41, 45], we are not aware of a similarly rigorous derivation of this key concept in prior work. In particular, we emphasize the separation between a random variable representing a response-time bound and its distribution (in contrast to most prior work), which later allows us to estimate a property of the distribution using samples of the random variable without needing to compute the distribution itself.

In the following, we build up to the notion of the probabilistic response-time bound $\mathcal{R}^\xi_{x,y}$ of a job $J_{x,y}$ in a given arrival sequence ξ (Def. 15 below). At a high level, the construction we describe is the following. Since the job $J_{x,y}$ under analysis is known, we can restrict our attention to the interval $[0, d_{x,y})$. We split this interval into two subintervals $[0, a_{x,y})$ and $[a_{x,y}, d_{x,y})$

(*i.e.*, at $J_{x,y}$'s arrival time $a_{x,y}$). First, we define the accumlated probabilistic carry-in at the end of $[0, a_{x,y})$. Second, given this carry-in at time $a_{x,y}$, we define a probabilistic response time bound of job $J_{x,y}$ only focusing on the interval $[a_{x,y}, d_{x,y})$.

When discussing random variables, it is common to drop the dependency on their sample spaces. That is, given a random variable $\mathcal{X} : \Omega \to \mathbb{T}$, it is customary to write $\mathcal{X}$ instead of $\mathcal{X}(\omega)$. For notational transparency, in the rest of *this section*, we do not follow this convention and explicitly specify the sample space.

We begin with a definition of an outcome space that can accommodate all the random variables needed to define $\mathcal{R}^{\xi}_{x,y}(\omega)$. Let $\Omega_{i,j}$ be the outcome set of the pWCET of a job $J_{i,j}$. Since ξ is known, we can compute the exact number of jobs of task τ_i arriving in the interval $[0, d_{x,y})$, defined as $m_i \triangleq |\xi_i([0, d_{x,y}))|$. The outcome set Ω_i of a random variable corresponding to the pWCETs of jobs of τ_i arriving in the interval $[0, d_{x,y})$ is thus the cross-product of the initial outcome sets, that is $\Omega_i \triangleq \Omega_{i,1} \times \ldots \times \Omega_{i,m_i}$. Similarly, we combine the per-task outcome sets to obtain an outcome set capable of accommodating the pWCETs corresponding to all jobs arriving in the interval $[0, d_{x,y})$, defined as $\Omega \triangleq \Omega_1 \times \ldots \times \Omega_n$.

We use $\Omega_{i,j}$ to refer to the j^{th} outcome space of the i^{th} task, and given an outcome $\omega \in \Omega$, analogously use $\omega_{i,j}$ to refer to the j^{th} outcome of the i^{th} task in ω (where $\omega_{i,j} \in \Omega_{i,j}$). Intuitively, every outcome $\omega \in \Omega$ corresponds to some *deterministic* evolution (or scenario) of the system from time 0 to time $d_{x,y}$. The corresponding event space 2^{Ω} and probability function, defined as the product of probabilities over the individual outcome sets $\Omega_{i,j}$, allow us to assign a notion of probability to such deterministic evolutions.

We next introduce probabilistic counterparts to the well-known basic building blocks of deterministic response-time analysis. We start with a probabilistic request-bound function.

Def. 11. The *probabilistic request-bound function* for a given task τ_i and interval $[t_1, t_2) \subseteq [0, d_{x,y})$ is a random variable that describes the workload of τ_i in $[t_1, t_2)$. Formally, for $\omega \in \Omega$: $\mathcal{RBF}^{\xi}_i([t_1, t_2), \omega) \triangleq \sum \{\mathcal{C}_{i,j}(\omega_{i,j}) \mid J_{i,j} \in \xi_i([t_1, t_2))\}$.

Def. 11 upper-bounds a task's *actual* demand since job costs are modeled using pWCETs (and not pETs) and because jobs are assumed to contribute their full cost upon arrival (even if they are later aborted). Building on $\mathcal{RBF}^{\xi}_i$, we next bound interference due to tasks with higher or equal priority.

Def. 12. The *probabilistic higher-or-equal-priority workload* with respect to task τ_i in a given interval $[t_1, t_2) \subseteq [0, d_{x,y})$ is a random variable that describes the total cost of all higher-or-equal-priority jobs released in $[t_1, t_2)$. Formally, for $\omega \in \Omega$: $\mathcal{W}^{\xi}_i([t_1, t_2), \omega) \triangleq \sum \left\{ \mathcal{RBF}^{\xi}_h([t_1, t_2), \omega) \mid \tau_h : \pi_h \succeq \pi_i \right\}$.

With $\mathcal{W}^{\xi}_i([t_1, t_2), \omega)$ in place, we are ready to define the well-known fixpoint at the heart of response-time analysis.

Def. 13. The *probabilistic fixpoint* for a given task τ_i, time instant $t < d_{x,y}$, and carry-in of size c is a random variable that describes the solution of the workload-fixpoint equation for an interval starting at time t with carry-in c. Formally: $\mathcal{Z}^{\xi}_i(c, t, \omega) \triangleq \min(\{d_{x,y} + \gamma\} \cup F)$, where $F \triangleq \left\{ t + \Delta \;\middle|\; \Delta > 0 : c + \mathcal{W}^{\xi}_i([t, t + \Delta), \omega) \leq \Delta \right\}$.

The artificial constant $d_{x,y} + \gamma$ encodes the occurrence of a deadline miss and upper-bounds the search range, as it ensures that any fixpoint solutions past the job's deadline (or the possibility that there are no solutions) are irrelevant.

Using the notion of a fixpoint, we define the carry-in at time t_c. Note that the carry-in at time t_c *does not* include the workload that arrives *at* time t_c, and recall that a *quiet time* occurs at a time t if any higher-or-equal-priority job that arrived before time t is completed by time t.

Def. 14. The *probabilistic carry-in* for a given task τ_i and time $t_c < d_{x,y}$ is a random variable that describes the unfinished interfering workload at time t_c. Formally, let $\mathcal{T}^{\xi}_i(t_c, \omega) \triangleq \min \left\{ t \;\middle|\; t \in [0, t_c] : \mathcal{Z}^{\xi}_i(0, t, \omega) > t_c \right\}$ be the latest quiet time (w.r.t. task τ_i) no later than t_c, then the carry-in at time t_c is $\mathcal{V}^{\xi}_i(t_c, \omega) \triangleq \mathcal{W}^{\xi}_i\big([\mathcal{T}^{\xi}_i(t_c, \omega), t_c), \omega\big) - (t_c - \mathcal{T}^{\xi}_i(t_c, \omega))$.

Finally, we arrive at a probabilistic response-time bound.

Def. 15. Job $J_{x,y}$'s *probabilistic response-time bound* (pRT) is the random variable $\mathcal{R}^{\xi}_{x,y}(\omega) \triangleq \mathcal{Z}^{\xi}_x(\mathcal{V}^{\xi}_x(a_{x,y}, \omega), a_{x,y}, \omega)$.

Intuitively, $\mathcal{R}^{\xi}_{x,y}(\omega)$ describes the probability that the response time of $J_{x,y}$ with respect to arrival sequence ξ is equal to a given value. However, due to the "stop value" in Def. 13, the distribution of $\mathcal{R}^{\xi}_{x,y}(\omega)$ is truncated at $d_{x,y} + \gamma$, so that the intuition holds only for values not exceeding $J_{x,y}$'s deadline. If $J_{x,y}$ misses its deadline (that is, Δ exceeds D_x), according to Def. 13, $\mathcal{R}^{\xi}_{x,y}(\omega)$ is always equal to $d_{x,y} + \gamma$, which suffices to define the job's probability of missing its deadline.

Def. 16. The *deadline-failure probability* (DFP) of a job $J_{x,y}$ in arrival sequence ξ is defined as $\mathbb{P}\left[\{\omega \mid \mathcal{R}^{\xi}_{x,y}(\omega) > D_x\}\right]$.

IV. Monte Carlo Estimation

In this section, we present our method for estimating the probabilistic response-time bound of a given job. Due to the challenges discussed in Section I, we relax the problem and allow the algorithm to **(i)** fail with a small, quantifiable probability and **(ii)** output only an approximate solution, where the permissible degree of approximation can be specified.

Problem statement. Given a task set τ, a task $\tau_x \in \tau$, an arrival sequence ξ, a job $J_{x,y}$ of task τ_x, the required *accuracy* $\delta > 0$, and the allowed *misestimation probability* $\varepsilon > 0$, derive estimates l and r for the unknown DFP $p = \mathbb{P}\left[\mathcal{R}^{\xi}_{x,y} > D_x\right]$ such that

$$|l - r| < \delta \text{ and } \mathbb{P}\left[l \leq p \leq r\right] \geq 1 - \varepsilon.$$

The *misestimation probability* has the frequentist interpretation that, among m runs of the algorithm, we expect *on average* $\varepsilon \cdot m$ runs to result in l and r such that $p \notin [l, r]$.

First, we explain why the reformulation of the problem still has practical significance. Indeed, usually we are not interested in obtaining the distribution of $\mathcal{R}^{\xi}_{x,y}$ *per se*. Rather, the relevant information is the DFP, that is $\mathbb{P}\left[\mathcal{R}^{\xi}_{x,y} > D_x\right]$. Further, the

Algorithm 1: DFP estimation

Input: τ, τ_x, ξ, δ, and ε.
Output: Estimate of $\mathbb{P}\left[\mathcal{R}^{\xi}_{x,y} > D_x\right]$.

1 $k \coloneqq 0$, $z \coloneqq \Phi^{-1}\left(1 - \frac{\varepsilon}{2}\right)$, $s \coloneqq \lceil (z/\delta)^2 \rceil$;
2 **for** 1 **to** s **do**
3 Draw sample via $\mathsf{S}^{\xi}_{x,y}$ (Def. 20 in Section V);
4 **if** $\mathsf{S}^{\xi}_{x,y} > D_x$ **then**
5 $k \coloneqq k + 1$;
6 $\tilde{s} \coloneqq s + z^2$, $\tilde{p} \coloneqq \frac{1}{\tilde{s}}\left(k + \frac{z^2}{2}\right)$;
7 **return** $\tilde{p} \pm z\sqrt{\frac{\tilde{p}(1-\tilde{p})}{\tilde{s}}}$

DFP estimate needs to be neither exact nor absolutely certain. For instance, if one seeks to verify that the DFP is at most 10^{-6}, one can use an inexact result, as long as the probability of the result being wrong is sufficiently small as to be negligible in the given engineering context (*e.g.*, 10^{-9}) and the upper bound on the estimated result does not exceed 10^{-6}, or in formal terms: $l \leq p \leq r \leq 10^{-6}$ with probability at least $1 - 10^{-9}$.

The above relaxations allow us to explore only a small fraction of the possible scenarios and then apply statistical generalization to estimate the actual probability of a deadline miss.

Estimation algorithm. To start, note that the expression $\mathcal{R}^{\xi}_{x,y} > D_x$ is itself a random variable. The inequality might resolve to a success if $\mathcal{R}^{\xi}_{x,y}$ is greater than D_x, or to a failure otherwise—it is a Bernoulli trial. Given a sequence of Bernoulli trials, the success probability can be estimated using binomial proportion confidence intervals, which we do in the following.

For ease of explanation, in this section we assume access to a black-box sample generator $\mathsf{S}^{\xi}_{x,y}$ (later defined in Section V) that provides samples with the same distribution as the random variable $\mathcal{R}^{\xi}_{x,y}$ (that is, $F[\mathcal{R}^{\xi}_{x,y}] = F[\mathsf{S}^{\xi}_{x,y}]$). Given $\mathsf{S}^{\xi}_{x,y}$, the problem reduces to collecting a sufficient number of samples to make statistical generalizations with the desired accuracy and misestimation probability, as shown in Algorithm 1.

Algorithm 1 samples a large number of values using $\mathsf{S}^{\xi}_{x,y}$, counts the number of outcomes for which $\mathsf{S}^{\xi}_{x,y}$ turned out to be larger than D_x, and then applies the Agresti-Coull confidence interval (Theorem 9) to estimate the ground-truth probability of observing $\mathsf{S}^{\xi}_{x,y} > D_x$. Since both $\mathsf{S}^{\xi}_{x,y}$ and $\mathcal{R}^{\xi}_{x,y}$ have the same distribution, this gives us a DFP estimate for a job $J_{x,y}$ in context of the given arrival sequence ξ.

To determine the number of samples s needed to achieve the target accuracy and error probability, Algorithm 1 computes $z \triangleq \Phi^{-1}\left(1 - \frac{\varepsilon}{2}\right)$, where Φ^{-1} is the ICDF of the standard normal distribution (*i.e.*, with mean 0 and variance 1) and selects $s \geq \left(\frac{z}{\delta}\right)^2$. The algorithm then draws s samples from $\mathsf{S}^{\xi}_{x,y}$ while counting the number of successes k, that is, the number of outcomes in which $\mathsf{S}^{\xi}_{x,y} > D_x$. Finally, since k is a binomial random variable, Algorithm 1 computes the Agresti–Coull confidence interval (Theorem 9) to determine l and r. We next argue that this statistical generalization is justified.

Theorem 17. *Let $\tau, \tau_x, \xi, \delta$, and ε be the input to Algorithm 1 and let (l, r) be the estimate that the algorithm produces. Then $l \leq \mathbb{P}\left[\mathcal{R}^{\xi}_{x,y} > D_x\right] \leq r$ with probability $1 - \varepsilon$ and $r - l \leq \delta$.*

Proof. Under the assumption that $F[\mathcal{R}^{\xi}_{x,y}] = F[\mathsf{S}^{\xi}_{x,y}]$, it is sufficient to estimate the probability $p = \mathbb{P}\left[\mathsf{S}^{\xi}_{x,y} > D_x\right]$ with accuracy at least δ and misestimation probability at most ε.

Let $\{r_1, \ldots, r_s\}$ be a sample of size s obtained by drawing from $\mathsf{S}^{\xi}_{x,y}$ (as in Lines 2–5 of Algorithm 1), and let $k \triangleq \sum_{j=1}^{s} [\![r_j > D_x]\!]$. The summands $[\![r_j > D_x]\!]$ are independent and identically distributed Bernoulli trials; k is thus a binomial random variable. Let l and r denote the lower and the upper bounds returned by Algorithm 1, respectively. Applying the Agresti-Coull [2] confidence interval (Theorem 9), we have that $p \in [l, r]$ with probability $1 - \varepsilon$. Further, the length of the returned interval is at most: $r - l = 2z\sqrt{\frac{\tilde{p}(1-\tilde{p})}{\tilde{s}}} \leq 2z\sqrt{\frac{1}{4\tilde{s}}} \leq 2z\sqrt{\frac{1}{4s}} \leq 2z\sqrt{\frac{\delta^2}{4z^2}} = 2z\frac{\delta}{2z} = \delta$. ∎

In the above proof, we use the coarse bound $\tilde{p}(1-\tilde{p}) \leq 1/4$. However, if the ground-truth probability p is low (as it usually is in practice) and the number of samples s is sufficiently large, then $\tilde{p}(1 - \tilde{p})$ becomes *much* smaller than $1/4$. As we show in the evaluation (Section VII-C), this means the required accuracy is often reached already with significantly fewer samples. Conversely, usually a much better accuracy than δ is achieved with s samples.

In place of the Agresti-Coull confidence interval, one could also use other confidence intervals such as the Wilson score interval [59] or the Clopper–Pearson interval [14], or concentration bounds such as Chernoff's bound [13] and Hoeffding's bound [26]. We use the Agresti-Coull confidence interval since it achieves a favorable trade-off between the probability to cover the ground-truth value and the length of the confidence interval for large samples [9].

V. Response-Time Bound Sampling

In this section, we define the sample generator $\mathsf{S}^{\xi}_{x,y}$ upon which Section IV rests. For Algorithm 1 to be practical, we must generate samples that are distributed as $\mathcal{R}^{\xi}_{x,y}$ (*i.e.*, $F[\mathcal{R}^{\xi}_{x,y}] = F[\mathsf{S}^{\xi}_{x,y}]$) in a sufficiently efficient manner. Fortunately, such a method exists: a simple simulation of the schedule together with inverse transform sampling does the trick.

We present this approach and argue its correctness in three steps. First, in Section V-A, we describe a simulation algorithm to compute the value of $\mathcal{R}^{\xi}_{x,y}$ assuming that job costs are fixed. Second, in Section V-B, we show that such an algorithm, if it is applied on a sequence of job costs randomly drawn from the pWCET distributions, yields response times that are distributed identically to $\mathcal{R}^{\xi}_{x,y}$. Finally, in Section V-C, we complete the sample generator by incorporating inverse transform sampling to facilitate the random generation of execution-time samples.

A. Simulation Algorithm

We start by introducing a simple algorithm that, assuming that all job costs are fixed, simulates the schedule in interval $[0, d_{x,y})$ and returns the response time of job $J_{x,y}$.

Algorithm 2: Schedule simulation $A^{\xi}_{x,y}$

Input: τ, ξ, $\vec{c}$, and $J_{x,y}$.
Output: Response time of $J_{x,y}$ if it meets its deadline or $d_{i,j} + \gamma$ in case of a deadline miss.

1 $v_0 \coloneqq 0$;
2 **for** $s \coloneqq 1$ **to** a **do**
3 $\quad w_{s-1} \coloneqq \sum \{c_{i,j} \mid J_{i,h} \in \xi_{\succeq x}(\ell_{s-1})\}$;
4 $\quad v_s \coloneqq \max\{0, v_{s-1} + w_{s-1} - (\ell_s - \ell_{s-1})\}$;
5 **for** $s \coloneqq a+1$ **to** d **do**
6 $\quad w \coloneqq \sum \{c_{i,j} \mid J_{i,j} \in \xi_{\succeq x}([\ell_a, \ell_s)\}$;
7 $\quad$ **if** $v_a + w \leq \ell_s - \ell_a$ **then**
8 $\quad\quad$ **return** $v_a + w$
9 **return** $d_{i,j} + \gamma$

For brevity, we first introduce a few auxiliary definitions. Recall that, since we analyze job $J_{x,y}$, we are interested in the interval $[0, d_{x,y})$. Let $m_i \triangleq |\xi_i([0, d_{x,y}))|$ be the number of jobs of task τ_i that arrive in the interval $[0, d_{x,y})$. Let $\vec{c}$ be a two-dimensional vector of job costs, where an element $c_{i,j}$ denotes the cost of job $J_{i,j}$. ($\vec{c}$ is not necessarily a rectangular matrix because each task may have a different number of jobs.) Finally, let $\ell \triangleq \{t \mid t \in [0, d_{x,y}) : \xi(t) \neq \varnothing\} \cup \{d_{x,y}\}$ be the union of **(i)** the set of time instants at which at least one job arrives and **(ii)** the deadline of the job $J_{x,y}$ under analysis. We assume ℓ to be sorted in increasing order and let ℓ_i denote the i^{th} element of the sequence. We also let a and d be the indices of respectively $a_{x,y}$ and $d_{x,y}$ in ℓ (*i.e.*, $\ell_a = a_{x,y}$ and $\ell_d = d_{x,y}$).

Algorithm 2 defines the simulation algorithm $A^{\xi}_{x,y}(\vec{c})$, which receives a vector of job costs $\vec{c}$ and outputs the response time of job $J_{x,y}$ in arrival sequence ξ. Conceptually, $A^{\xi}_{x,y}$ is just a straight-forward deterministic simulation of the schedule for the given arrival times and job costs. First, the amount of the carry-in v_a at time ℓ_a is computed (Lines 2–4). For ℓ_0, the carry-in is 0. For $\ell_0 < \ell_s \leq \ell_a$, the carry-in can be computed recurrently based on the carry-in v_{s-1} at time ℓ_{s-1}, the workload w_{s-1} released at time ℓ_{s-1}, and the time difference between ℓ_{s-1} and ℓ_s as $v_s \triangleq \max\{0, v_{s-1} + w_{s-1} - (\ell_s - \ell_{s-1})\}$.

Afterwards (lines Lines 5–8) the algorithm searches for a time $\ell_s \leq \ell_d$ such that the carry-in at time ℓ_a and the new workload released in $[\ell_a, \ell_s)$, $w \triangleq \sum \{c_{i,j} \mid J_{i,j} \in \xi_{\succeq x}([\ell_a, \ell_s)\}$, are fully consumed (*i.e.*, $v_a + w \leq \ell_s - \ell_a$). If such a time ℓ_s is found, then the algorithm returns the response time (*i.e.*, $v_a + w$). Otherwise, not all higher-or-equal-priority workload is consumed by the job's deadline. Hence, the simulation indicates that the response time exceeds the deadline by returning $d_{x,y} + \gamma$ in Line 9 (recall that γ is the smallest unit of time).

B. Sampling via Algorithm 2

In this subsection, we argue that if, instead of a vector of fixed job costs, Algorithm 2 is run on a vector of pWCETs (*i.e.*, random variables), then the probability of $A^{\xi}_{x,y}$ yielding some value r is equal to the probability of $\mathcal{R}^{\xi}_{x,y}$ being equal to r.

First, we prove an auxiliary lemma stating that $A^{\xi}_{x,y}$'s output agrees with the response-time bound $\mathcal{R}^{\xi}_{x,y}$ given in Def. 15. To state the claim formally, suppose we have some collection of job costs $\vec{c}$ and an outcome $\omega \in \Omega$ such that $\mathcal{C}_{i,j}(\omega_{i,j}) = c_{i,j}$ for each $J_{i,j} \in \xi([0, d_{x,y}))$. We observe that in this case the values of $A^{\xi}_{x,y}(\vec{c})$ and $\mathcal{R}^{\xi}_{x,y}(\omega)$ are the same.

Lemma 18. *Given a vector of job costs $\vec{c}$ and an outcome $\omega \in \Omega$, if $\mathcal{C}_{i,j}(\omega_{i,j}) = c_{i,j}$ for each $J_{i,j} \in \xi([0, d_{x,y}))$, then $A^{\xi}_{x,y}(\vec{c}) = \mathcal{R}^{\xi}_{x,y}(\omega)$.*

Proof. We carry out the proof in three steps. First, we show that, if $\ell_i \leq \ell_a$ is the last time before ℓ_a with $v_i = 0$, then ℓ_i is also the last quiet time in the sense of Def. 14. Second, we show that Lines 1-4 indeed compute the carry-in at time ℓ_a (Def. 14). Third, we show that Lines 5-9 compute $\mathcal{R}^{\xi}_{x,y}$.

Let $\ell_q \leq \ell_a$ be the last instant with carry-in 0, that is $v_q = 0$ and $\forall q' \in \{q+1, \ldots, a\} : v_{q'} > 0$. This is equivalent to the conjunction of **(i)** $\mathcal{Z}^{\xi}_x(0, \ell_q, \omega) > \ell_a$, since all $v_{q'}$ following v_q are positive, and **(ii)** $\forall i \in \{1, \ldots, q-1\} : \mathcal{Z}^{\xi}_x(0, \ell_i, \omega) \leq \ell_q$, since no fixpoint iteration that starts before ℓ_q can exceed time ℓ_q because $v_q = 0$. The conjunction of (i) and (ii) is equivalent to the fact that $\mathcal{T}^{\xi}_x(\ell_a, \omega) = \ell_q$ (recall Def. 14).

We next show that $\mathcal{V}^{\xi}_x(\ell_a, \omega) = v_a$: since $v_q = 0$ and $\forall q' \in \{q+1, \ldots, a\} : v_{q'} > 0$, we can replace v_a with $\sum_{i=q}^{a-1} w_i - (\ell_a - \ell_q)$, which equals $\mathcal{W}^{\xi}_x([\ell_q, \ell_a), \omega) - (\ell_a - \ell_q)$, since $w_i = \mathcal{W}^{\xi}_x([\ell_i, \ell_{i+1}), \omega)$ and $\mathcal{T}^{\xi}_x(\ell_a, \omega) = \ell_q$. Hence $v_a = \mathcal{V}^{\xi}_x(\ell_a, \omega)$.

It remains to be shown that $v_a + w = \mathcal{R}^{\xi}_x(\omega)$. Recall from Def. 15 that $\mathcal{R}^{\xi}_x(\omega) = \mathcal{Z}^{\xi}_x(\mathcal{V}^{\xi}_x(\ell_a, \omega), \ell_a, \omega)$, where $\mathcal{Z}^{\xi}_x(\mathcal{V}^{\xi}_x(\ell_a, \omega), \ell_a, \omega)$ searches for the least time instant strictly greater than ℓ_a when both the carry-in $\mathcal{V}^{\xi}_x(\ell_a, \omega)$ and the higher-or-equal-priority workload $\mathcal{W}^{\xi}_x([\ell_a, \ell_s), \omega)$ are consumed. This is exactly the process performed in Lines 5-8, since the carry-in is $v_a = \mathcal{V}^{\xi}_x(\ell_a, \omega)$ and the higher-or-equal-priority workload is $w = \mathcal{W}^{\xi}_x([\ell_a, \ell_s), \omega)$. Hence the algorithm indeed searches for an ℓ_s such that $\mathcal{V}^{\xi}_x(\ell_a, \omega) + \mathcal{W}^{\xi}_x([\ell_a, \ell_s), \omega) = v_a + w \leq \ell_s - \ell_a$.

Furthermore, $\mathcal{R}^{\xi}_x(\omega) = d_{x,y} + \gamma$ iff there is no fixpoint at or before $d_{x,y}$ (recall $\mathcal{Z}^{\xi}_x(\mathcal{V}^{\xi}_x(\ell_a, \omega), \ell_a, \omega)$). This, in turn, is equivalent to the fact that the algorithm fails to find $\ell_s \leq \ell_d$ such that $v_a + w \leq \ell_s - \ell_a$; thus the algorithm returns $d_{x,y} + \gamma$ in Line 9. Therefore, the claimed equivalence follows. ∎

Lemma 18 above is a claim about a (fixed) vector of *constants* representing job costs, $\vec{c}$, related to a particular element of the outcome space $\omega \in \Omega$. To formally make a claim about the *distribution of outputs of* $A^{\xi}_{x,y}$ if invoked repeatedly on inputs drawn from pWCET distributions, we need to lift the established equivalence to the level of random variables.

To this end, let $\vec{\mathcal{C}}$ denote a two-dimensional vector of pWCETs, such that $\mathcal{C}_{i,j}$ is the pWCET of job $J_{i,j}$ (Def. 10) for each $J_{i,j} \in \xi([0, d_{x,y}))$. We can interpret this vector as a random variable that maps each element of the outcome set $\omega \in \Omega$ to a two-dimensional vector of job costs $\vec{c}$, with $c_{i,j} = \mathcal{C}_{i,j}(\omega_{i,j})$. Taking this interpretation one step further, a function $A^{\xi}_{x,y}(\vec{\mathcal{C}})$ that applies $A^{\xi}_{x,y}$ to the vector of job costs produced by $\vec{\mathcal{C}}$ is itself a random variable that maps each element of the outcome set $\omega \in \Omega$ to a scalar, namely the output of Algorithm 2. With this setup in place, we can formally relate the output probability of $A^{\xi}_{x,y}$ with $\mathcal{R}^{\xi}_{x,y}$.

Lemma 19. *For any constant $r \in \mathbb{T}$, with respect to the outcome set Ω, $\mathbb{P}\left[A^{\xi}_{x,y}(\vec{\mathcal{C}}) = r\right] = \mathbb{P}\left[\mathcal{R}^{\xi}_{x,y} = r\right]$.*

Proof. It suffices to show that $E_1 \triangleq \left\{\omega \in \Omega \;\middle|\; \mathcal{R}^{\xi}_{x,y}(\omega) = r\right\}$ is equal to $E_2 \triangleq \left\{\omega \in \Omega \;\middle|\; A^{\xi}_{x,y}(\vec{\mathcal{C}}(\omega)) = r\right\}$. We show that $\omega \in E_1$ if and only if $\omega \in E_2$ for an arbitrary $\omega \in \Omega$. Given $\omega \in \Omega$, $\vec{\mathcal{C}}$ yields a vector $\vec{c}$ such that $\mathcal{C}_{i,j}(\omega_{i,j}) = c_{i,j}$ for all i and j, which is used as input for $A^{\xi}_{x,y}$. Hence, it remains to be shown that $A^{\xi}_{x,y}(\vec{c}) = \mathcal{R}^{\xi}_{x,y}(\omega)$, which holds according to Lemma 18. Therefore $\omega \in E_1 \iff \omega \in E_2$. ■

C. Sampling Job Costs

In principle, one could use Algorithm 2 and Lemma 19 to sample values from $\mathcal{R}^{\xi}_{x,y}$. That, however, would require a method for sampling uniformly at random from the underlying outcome set Ω, which conceptually represents the set of possible *system evolutions*. Sampling Ω directly would thus pose a multitude of challenges, as it is necessarily workload-specific (*e.g.*, it encodes control-flow information such as the branches taken by each job) and in all likelihood extremely complicated to represent and manipulate. In other words, Ω is a key abstraction, but not a practical building block. To bypass the dependency on Ω, we employ inverse transform sampling, which allows us to sample job costs directly without a need to construct or even consider the underlying outcome set Ω.

Let $\vec{\mathcal{U}}$ be a two-dimensional vector of uniform random variables ranging over the interval $[0, 1]$ with the same shape as vector $\vec{\mathcal{C}}$. We use $\mathcal{U}_{i,j}$ to refer to the j^{th} element of the i^{th} row of $\vec{\mathcal{U}}$. Furthermore, let $Q(\vec{\mathcal{U}})$ denote a two-dimensional vector obtained by applying task τ_i's ICDF Q_i (Def. 6) to every element of the i^{th} row of $\vec{\mathcal{U}}$. In other words, an element of $Q(\vec{\mathcal{U}})$ with indices i and j is equal to $Q_i(\mathcal{U}_{i,j})$.

The basic idea is to apply $A^{\xi}_{x,y}$ to $Q(\vec{\mathcal{U}})$ instead of $\vec{\mathcal{C}}$. As we show in the following, the output distribution of $A^{\xi}_{x,y}$ does not change as a result. The benefit of switching from $\vec{\mathcal{C}}$ to $Q(\vec{\mathcal{U}})$ is that there exists a variety of readily available and fast methods to sample from $\mathcal{U}([0, 1])$ (*e.g.*, [28, 30, 58]), and by extension thus also from $Q(\vec{\mathcal{U}})$. Hence, we define $\mathsf{S}^{\xi}_{x,y}$ as follows.

Def. 20. $\mathsf{S}^{\xi}_{x,y} \triangleq A^{\xi}_{x,y}(Q(\vec{\mathcal{U}}))$.

We now establish the defining property of $\mathsf{S}^{\xi}_{x,y}$: it has the same distribution as $\mathcal{R}^{\xi}_{x,y}$. We first note that pWCET samples and samples drawn from $Q(\vec{\mathcal{U}})$ are identically distributed.

Lemma 21. *For any $\mathcal{C}_{i,j}$ and any $a \in \mathbb{T}$, it holds that $\mathbb{P}[\mathcal{C}_{i,j} = a] = \mathbb{P}[Q_i(\mathcal{U}_{i,j}) = a]$.*

Proof. By Fact 4, it is sufficient to prove that the distribution of $\mathcal{C}_{i,j}$ is equal to the distribution of $Q_i(\mathcal{U}_{i,j})$. By construction, $\mathcal{U}_{i,j}$ is distributed uniformly in $[0, 1]$. Thus, by Theorem 7, the random variable $Q_i(\mathcal{U}_{i,j})$ has the distribution $F_i = F[\mathcal{C}_{i,j}]$. ■

Finally, we show that $\mathsf{S}^{\xi}_{x,y}$ combines job-cost samples in such a way that the initial distribution is preserved.

Theorem 22. $F[\mathcal{R}^{\xi}_{x,y}] = F[\mathsf{S}^{\xi}_{x,y}]$.

Proof. By Fact 4, it suffices to show that $\mathbb{P}\left[\mathcal{R}^{\xi}_{x,y} = r\right]$ is equal to $\mathbb{P}\left[\mathsf{S}^{\xi}_{x,y} = r\right]$ for an arbitrary $r \in \mathbb{T}$.

$$\begin{aligned}
\mathbb{P}\left[\mathcal{R}^{\xi}_{x,y} = r\right] &\overset{(1)}{=} \mathbb{P}\left[A^{\xi}_{x,y}(\vec{\mathcal{C}}) = r\right] \\
&\overset{(2)}{=} \sum_{c_{1,1},\ldots,c_{n,m_n}} \left[\!\left[A^{\xi}_{x,y}(\vec{c}) = r\right]\!\right] \prod_{i,j} \mathbb{P}\left[\mathcal{C}_{i,j} = c_{i,j}\right] \\
&\overset{(3)}{=} \sum_{c_{1,1},\ldots,c_{n,m_n}} \left[\!\left[A^{\xi}_{x,y}(\vec{c}) = r\right]\!\right] \prod_{i,j} \mathbb{P}\left[Q_i(\mathcal{U}_{i,j}) = c_{i,j}\right] \\
&\overset{(4)}{=} \mathbb{P}\left[\mathsf{S}^{\xi}_{x,y} = r\right],
\end{aligned}$$

where *(1)* holds by Lemma 19, *(2)* holds by Fact 8, *(3)* holds by Lemma 21, and *(4)* holds by Fact 8 and Def. 20. ■

A simple corollary of Theorem 22 is that we can use $\mathsf{S}^{\xi}_{x,y}$ to assess the probability of a deadline violation.

Corollary 23. $\mathbb{P}\left[\mathsf{S}^{\xi}_{x,y} > D_x\right] = \mathbb{P}\left[\mathcal{R}^{\xi}_{x,y} > D_x\right]$.

VI. Worst-Case Deadline Failure Probability

In this section, we explain how to put everything together and apply the algorithm presented in Sections IV and V to estimate the WCDFP. To this end, we first recall its definition and a theorem proven by Maxim and Cucu-Grosjean [41].

The WCDFP of task τ_i is an upper bound on the probability that any single job of the task misses its deadline [17].

Def. 24. The *worst-case deadline failure probability* (WCDFP) of a task τ_i is $\Lambda_i \triangleq \max_{\xi} \max_j \mathbb{P}\left[\mathcal{R}^{\xi}_{i,j} > D_i\right]$.

The DFP estimation algorithm (Algorithm 1) does not rely on any particular system properties (besides assuming a fully preemptive uniprocessor) and can estimate the DFP of a given job in a given arrival sequence independently of the kind of deadline constraints or structure of the arrival pattern. Furthermore, the estimated DFP remains sound regardless of a system's job abortion policy, since upon a job arrival Algorithm 2 (in Lines 3 and 6) accounts for the full job cost, which is a safe upper bound even if jobs are aborted [20].

Unfortunately, Def. 24 is not suitable for an actual computation of the WCDFP since it takes the maximum over an infinite set of arrival sequences and jobs. However, under some not too-restrictive assumptions, Maxim and Cucu-Grosjean [41] showed that the calculation of the WCDFP under fixed-priority scheduling can be reduced to one specific situation. Specifically, for a set of constrained-deadline sporadic tasks scheduled by a fully preemptive fixed-priority scheduler, if overruns are prevented by aborting any incomplete jobs at their deadline, it is sufficient to analyze the first job under a critical-instant pattern ξ_{crit}. (That is, the task under analysis and all higher-or-equal-priority tasks release a job simultaneously at time 0 and continue to release jobs with minimal inter-arrival time.)

Theorem 25 (Theorem 1 in [41])**.** *Given a fully preemptive fixed-priority scheduler, a set of constrained-deadline sporadic tasks, and under the assumption that incomplete jobs are aborted at their deadline, the first job in the synchronous busy period exhibits a task's WCDFP: $\Lambda_i = \mathbb{P}\left[\mathcal{R}^{\xi_{crit}}_{i,1} > D_i\right]$.*

Since our system model matches the system model studied by Maxim and Cucu-Grosjean [41], the WCDFP of task τ_x can be estimated by running Algorithm 2 for $J_{x,1}$ with $\xi = \xi_{\text{crit}}$.

VII. Evaluation

We conducted an empirical evaluation using synthetic workloads to asses the practicality of the proposed method and to compare its performance to state-of-the-art algorithms.

Setup. We randomly generated sets of sporadic tasks with two execution modes each, representing a task's *typical* behavior and a rare *exceptional mode*. For a given task-set size n and expected total utilization u, we drew n periods $p_1, \ldots, p_n$ uniformly at random from $\{1, 2, 5, 10, 20, 50, 100, 200, 500, 1000\}$ms, a set of periods commonly used in automotive systems [25, 29, 52, 56]. Next, we used the Dirichlet-Rescale algorithm [23, 24] to draw n utilization values $u_1, \ldots, u_n$ summing to u. The pWCET of each task τ_i was defined as follows: c with probability 0.95 (the typical mode), and $4c$ with probability 0.05 (the exceptional mode), where c was scaled such that the expected value of the pWCET was $u_i \cdot p_i$. Finally, we assigned rate-monotonic priorities.

We evaluated four algorithms: classic exact convolution with state merging **(CX)** by Maxim and Cucu-Grosjean [41]; convolution with *reduced-pessimism resampling* **(CR)** by Maxim et al. [43]; the analytical approach using Chernoff bounds **(AB)** by Chen and Chen [10]; and the Monte Carlo approach **(MC)** proposed in this paper. We also tested *k-most popular* [42] and *uniform resampling* [43] as alternatives to **CR** and analytical bounds using Hoeffding's and Bernstein's inequalities [57] as alternatives to **AB**, but do not report those results since they proved to be less precise than **CR** and **AB**, respectively.

To allow for meaningful runtime comparisons, all approaches were implemented in the same programming language, Rust, and deployed on the same machine equipped with two Intel Xeon "Platinum 8180" processors clocked at 2.50 GHz and 394 GiB RAM. To obtain a fair basis for comparison, we opted for a *single-threaded* implementation of all methods; the degree of parallelism offered by the hardware is hence irrelevant.

For convolution-based approaches, we implemented direct convolution (with quadratic time complexity) and used Rust's `HashMap` library for automatic state merging. We implemented the **AB** method's search for minimizing parameters using ternary search since the minimization problem is convex [12].

A. Comparison with State-of-the-Art Approaches

We compared the WCDFP estimates produced by **MC** and the baselines. We generated 2500 task sets, 50 for each combination of $u \in \{0.75, 0.8, 0.85, 0.9, 0.95\}$ and $n \in \{5, 10, \ldots, 50\}$, and let **AB**, **CR**, and **MC** estimate the WCDFP of the lowest-priority task in each task set. Under **CR**, distributions were sampled down to 2000 values whenever reaching 4000 values. For **MC**, we set the *misestimation probability* to $\varepsilon = 10^{-6}$ (*i.e.*, accepting one expected failure in 1 million runs) and the time budget to 8 minutes per task set (which was **CR**'s average runtime for the evaluated task sets).

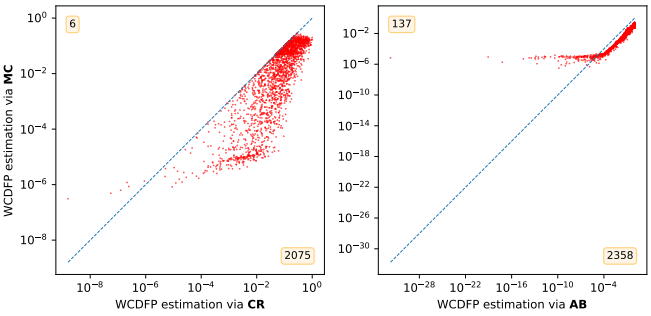


Fig. 2: Scatter plot of WCDFP estimates given by **CR** and **MC** (left plot) and **AB** and **MC** (right plot) for each of the generated task sets. A point under the blue diagonal line indicates a task set where **MC** provides a better result. The numeric labels in the top-left and bottom-right corners indicate the number of points above and below the line, respectively.

Fig. 2 shows the accuracy of **MC** in comparison to **CR** and **AB** as scatter plots. Each point corresponds to one of the 2500 generated task sets. Compared to **CR** (left plot), **MC** provided a better result for 2075 of the tested task sets, whereas **CR** was more accurate for only 6 task sets, for which it could determine WCDFPs below $\approx 10^{-6}$. (The remaining 419 points are on or very close to the diagonal line and hence inconclusive.) In comparison to **AB** (right plot), **MC** provided a lower WCDFP estimate for 2358 task sets. However, the plot also shows that, in the given 8-minute time budget, **MC** struggled to obtain a sufficient number of samples to derive WCDFPs lower than $\approx 10^{-6}$, which is not an issue for **AB**. As a result, **AB** provided better estimates for 137 task sets. Generally, Fig. 2 shows **MC** to be an attractive alternative to both **CR** and **AB**.

To assess the impact of **CR**'s resampling threshold, we also tested down-sampling to 250, 500, 1000, 2500, and 5000 retained values (when reaching twice the number of values). While increasing the number of retained values noticeably improved **CR**'s accuracy, it also had a major runtime impact (doubling the threshold roughly doubled the runtime). Consequently, even when retaining 5000 values, **MC** still clearly outperformed **CR**. Retaining more than 5000 values proved difficult, since for such large thresholds it took **CR** on average more than 30 minutes to analyze a single task set.

In the remainder of the paper, we summarize scatter plots as box plots of the *ratio* between the WCDFP as given by a baseline method and by **MC** due to space constraints. Using this method, Fig. 3 reports the same data as Fig. 2, grouped by either n or u. For instance, Fig. 3a shows the WCDFP obtained via **CR** divided by r (*i.e.*, the upper bound of the WCDFP interval reported by **MC**) for a varying n.

As evident in Fig. 3a, for smaller cardinalities (*i.e.*, $n \leq 15$), **MC** performs similar to the **CR** baseline, yielding sometimes better and sometimes worse estimates. However, as the number of tasks grows (and thus the number of jobs that must be considered), the ratio grows steadily to around 10^1–10^2, meaning that **MC** is one to two orders of magnitude more

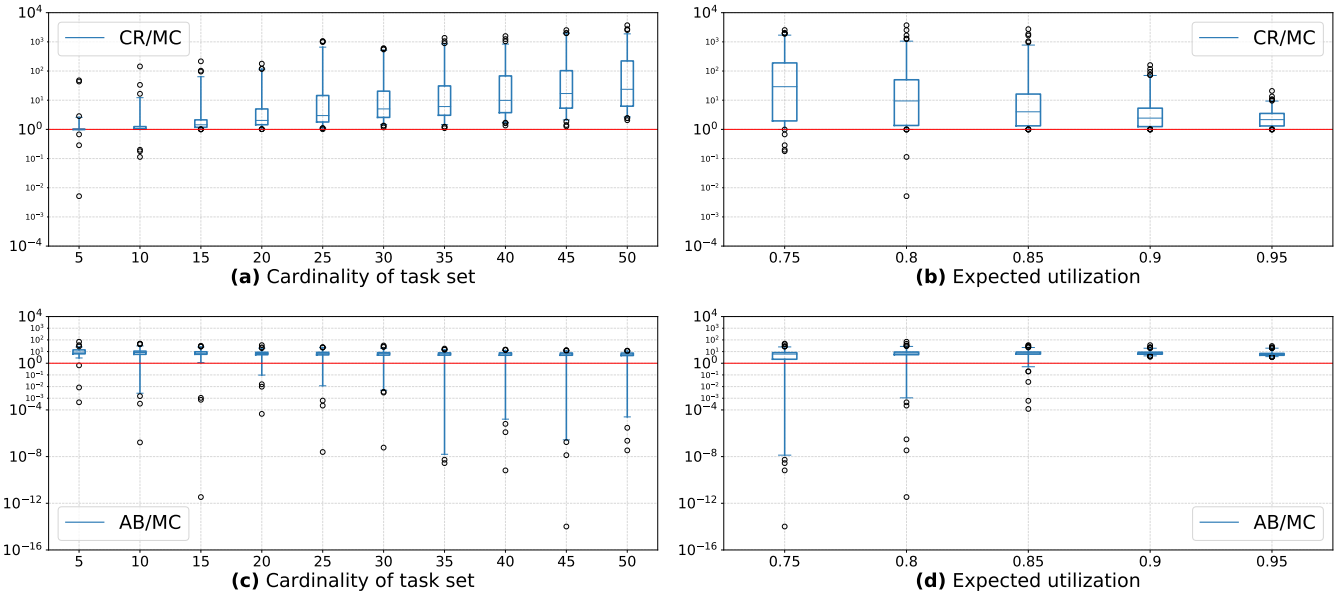


Fig. 3: Box plots of the ratios **CR** / **MC** (upper row) and **AB** / **MC** (lower row). A value exceeding 1 indicates **MC** yielding a lower (*i.e.*, better) WCDFP estimate than the respective comparison algorithm. Each box extends from the lower to the upper quartile, with a line at the median; the bottom and top whiskers indicate the 1^{st} and 99^{th} quantile, respectively.

precise than **CR** in these cases. The reason is that **MC** scales well with the number of jobs in the interval under analysis, and thus maintains roughly the same precision irrespective of n, whereas the number of resampling operations in **CR** grows with the number of jobs, each time accruing more pessimism.

Fig. 3b shows the effects of varying the expected total utilization. For $u = 0.75$, **MC** delivered results roughly one to two orders of magnitude better than **CR**. As the utilization increases, the magnitude of the ratios decreases, but **MC** remains preferable to **CR** throughout the range. For $u = 0.95$, the actual WCDFP is $\approx 10^{-1}$, at which point the normalization obscures the trend due to the large denominator (*e.g.*, even trivially reporting 1 would not register as a large ratio).

Figs. 3c and 3d show a larger variance in the comparison with **AB**. As both the median and the average are above 1, **MC** tends to yield preferable results for all considered n and u. The large variance stems from the fact that our sequential **MC** implementation can collect only a limited number of samples in the given 8-minute time budget, as already discussed in the context of Fig. 2. Therefore, if the actual WCDFP is vanishingly small, **MC** outputs conservative bounds dominated by the limited number of samples, whereas analytical upper-bounds have no such restrictions. Since the **AB** approach can be computed very quickly, this suggests that one should apply it first, and run the **MC** method with a sufficiently large time budget only if the **AB**-provided estimate is unsatisfying.

B. Influence of the pWCET Distribution

To assess how each task's pWCET parameter influences analysis runtime and accuracy, we varied the the shape of the generated pWCET distributions in three ways, as discussed next. For each considered pWCET shape, we generated 500 task sets, 10 for each combination of $u \in \{0.75, 0.8, 0.85, 0.9, 0.95\}$ and $n \in \{5, 10, \ldots, 50\}$. As before, we let **AB**, **CR**, and **MC** estimate the WCDFP of the lowest-priority task in each task set.

First, we varied the difference between normal- and exceptional-mode execution times. For each task, the pWCET was defined to be c with probability 0.95 and $k \cdot c$ with probability 0.05, where the *discrepancy* k varied across $\{2, 3, 4, 5, 8, 16\}$. The results are shown in Figs. 4a and 4d. The main observation is that **MC** remains preferable as the discrepancy parameter increases, thus showing that it is not negatively affected by more extreme pWCETs. For higher discrepancy values, the ratios stabilize around 10^1, which is due to the fact that higher-discrepancy task sets tend to exhibit higher WCDFPs, so that even a trivial upper bound of 1 would not register as a large ratio. Nonetheless, **MC** was more accurate than either **CR** or **AB** for virtually all task sets with $k \geq 5$.

Next, we varied how frequently jobs execute in normal mode by defining each task's pWECT to be c with probability p, and $4c$ with probability $1 - p$, where the *normal-mode probability* p varied across $\{0.5, 0.75, 0.9, 0.95, 0.99, 0.999\}$. The results shown in Figs. 4b and 4e reveal two main trends. If the normal-mode probability is relatively low, **AB** yields good bounds while the results of **CR** are poor: in some cases, **CR**'s estimates were five orders of magnitude worse than those provided by **MC**. In contrast, when the exceptional-mode probability is vanishingly small, **AB** becomes somewhat more pessimistic, while **CR** improves substantially: in some cases, **CR** yielded WCDFP estimates up to 12 orders of magnitude lower than **MC**. This observation is again explained by the employed sample budget, which limited **MC**'s ability to assess rare events.

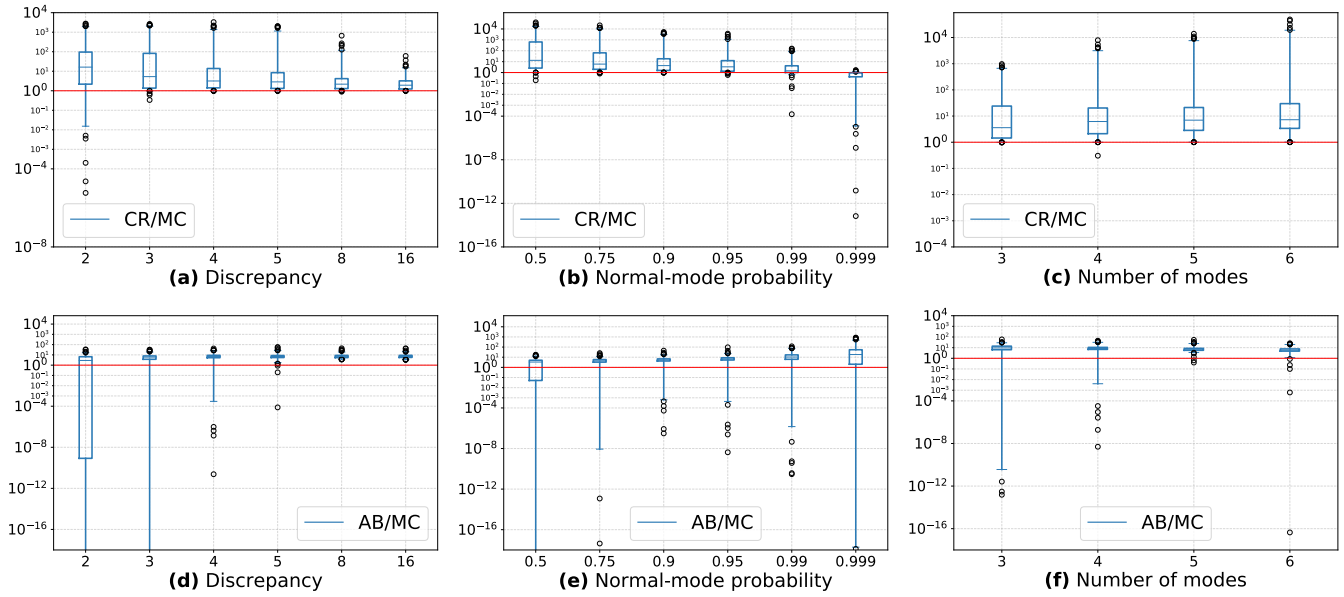


Fig. 4: Box plots of the ratios **CR** / **MC** (upper row) and **AB** / **MC** (lower row) grouped by the discrepancy (left column), probability of the normal mode (middle column), and the number of modes (right column).

Finally, we varied the number of execution modes. To this end, each task's pWCET was defined as follows: for a given *number of modes* $m \in \{3, 4, 5, 6\}$, mode $s \in \{2, \ldots, m\}$ had execution time $2(s-1) \cdot c$ and probability $2^{m-s}/100$, with the remaining probability assigned to the normal mode $s = 1$ with execution time c. The results are depicted in Figs. 4c and 4f, which show that **CR**'s accuracy relative to **MC** consistently degrades as the number of modes increases. **AB**'s relative accuracy stabilizes at a plateau around one order of magnitude worse than **MC**, with some outliers where **AB** yields results below **MC**'s resolution limit (for the given time budget).

Overall, we conclude that **MC**'s performance is robust with regard to changes in the shapes of the pWCET distributions.

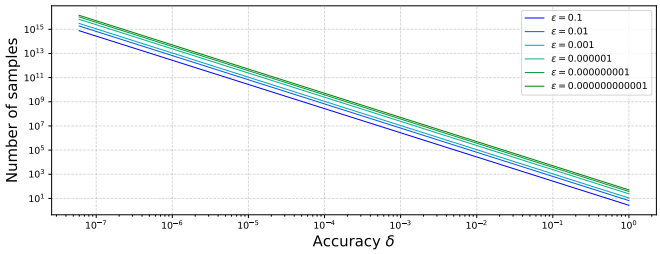


Fig. 5: Number of samples predicted to be needed in the worst case as a function of δ for various ε. The growth in the number of samples is largely driven by decreasing δ, with decreases in ε having a diminishing and comparably small effect.

C. Scalability of the Approach

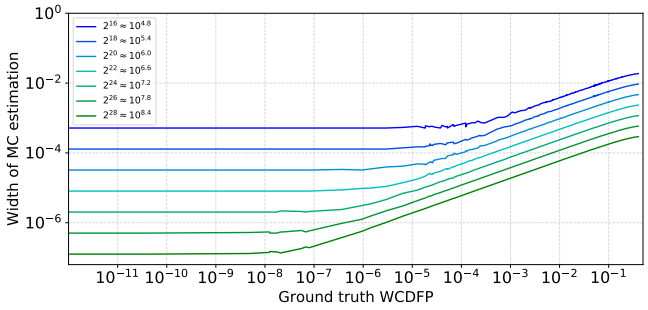
One might be worried that the precision grows too slowly with increasing sample counts. Indeed, as illustrated in Fig. 5, according to the inequality $s \geq (z/\delta)^2$, a precision of 10^{-6} would require $\approx 10^{12}$ (*i.e.*, $\approx 2^{40}$) samples, which is impractical. Luckily, as pointed out in Section IV, this is an artifact of the proof that upper-bounds the number of samples needed in the *worst-case scenario*, which corresponds to a ground-truth WCDFP of 0.5. However, in practice, one is usually interested in workloads with a WCDFP *much* lower than 10^{-1}.

To explore **MC**'s actual precision, we conducted an experiment in which we measured the attained precision as a function of the ground-truth WCDFP. To this end, we generated 1200 task sets in total, 80 task sets for each combination of $u \in \{0.55, 0.65, 0.75, 0.85, 0.95\}$ and $n \in \{2, 4, 6\}$. The setup was restricted to small task sets to enable computation of the exact WCDFP via **CX**, which does not scale much further.

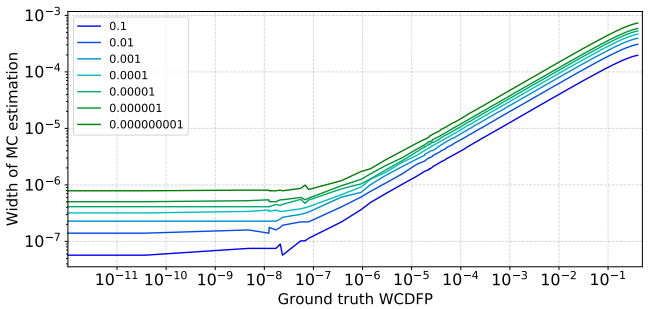
Each task set was evaluated with **CX** with a 30-minute timeout to determine the ground-truth WCDFP. Task sets that **CX** could not analyze within 30 minutes were removed from further consideration, resulting in 730 task sets with a known exact WCDFP (*i.e.*, 399, 253, and 78 task sets with 2, 4, and 6 tasks, respectively). Next, we evaluated these 730 task sets with **MC** (with $\varepsilon = 10^{-6}$) configured with different *sample budgets* $2^{16}, 2^{18}, \ldots, 2^{28}$ and measured the empirical width $|r - l|$ of the confidence intervals reported by Algorithm 1.

Fig. 6a shows the results. When the WCDFP is large (*e.g.*, 10^{-1}), even with $2^{28} \approx 10^{8.4}$ samples (the bottommost curve), **MC** cannot reach a modest estimation width of 10^{-4}. The reason is that, for large WCDFPs, the term $\tilde{p}(1-\tilde{p})$ in Line 7 of Algorithm 1 does not become substantially smaller than $1/4$.

However, when the ground-truth WCDFP decreases, $\tilde{p}(1-\tilde{p})$ decreases too, which shortens the width of the actually attained confidence intervals considerably. Hence, as Fig. 6a shows,



(a) Attained empirical width $|r - l|$ for fixed sample budgets



(b) Attained empirical width $|r - l|$ for various choices of ε

Fig. 6: Empirical width $|r - l|$ of the attained confidence interval in relation to the ground-truth WCDFP.

if the ground-truth WCDFP is below 10^{-6}, then $2^{28} \approx 10^{8.4}$ samples actually result in an accuracy of $|r - l| < 10^{-6}$ (in contrast to the 10^{12} worst-case prediction, recall Fig. 5).

Even though in practice we do not have *a priori* knowledge of the ground-truth WCDFP, the property highlighted in this subsection plays a big role in the choice of the number of samples. Usually, real-time systems are designed with some prior belief about the allowed WCDFP. If the WCDFP is expected to be small (*e.g.*, 10^{-6}) and the needed accuracy is $\delta = 10^{-6}$, then one can simply try generating in the order of $2^{28} \approx 10^{8.4}$ samples—most likely, **MC** will provide a satisfactory answer nonetheless. If however **MC** ultimately fails to provide an answer with the required accuracy (*i.e.*, if it reports an interval length exceeding δ), then it is a reasonable guess to assume that the ground truth actually exceeds 10^{-6}.

Finally, Fig. 6a also confirms an effect already discussed in the context of Figs. 2, 3c, and 3d: if **MC** is limited to a relatively small number of samples, such as $2^{16} \approx 10^{4.8}$ (the topmost curve), then it cannot produce estimates below a certain threshold dependent on the sample budget, which manifests in Fig. 6a as the flat plateaus of the curves extending to the left.

D. Choice of Misestimation Probability ε

We next conducted an experiment to assess the influence of the misestimation probability ε on the empirical width of the attained confidence interval. Intuitively, it stands to reason that, to increase confidence in the result, one should be more conservative in one's predictions. Thus, as ε decreases, we can expect the width of the confidence interval to grow. However, it is far from obvious *how large* an effect a decrease in ε has.

To measure this effect, we reused the 730 workloads with known exact WCDFPs already discussed in Section VII-C and ran **MC** with a fixed sample budget of 2^{26} and $\varepsilon \in \{10^{-1}, \ldots, 10^{-6}, 10^{-9}\}$. Fig. 6b depicts the results.

Similarly to Fig. 6a, the width of the interval decreases with the ground-truth WCDFP. However, what may be surprising is that, when the allowed misestimation probability is decreased from $\varepsilon = 10^{-2}$ (*i.e.*, one misestimate in *one hundred* runs) to $\varepsilon = 10^{-9}$ (*i.e.*, one misestimate in *one billion* runs), the empirical width of the attained confidence intervals increases by only a factor of 10. The reason is that ε influences the width of the interval only though the quantile function $z \triangleq \Phi^{-1}\left(1 - \frac{\varepsilon}{2}\right)$ in Theorem 9, which grows very slowly as ε tends to 0.

There is thus little harm in picking a suitably low misestimation probability ε that can be considered negligible.

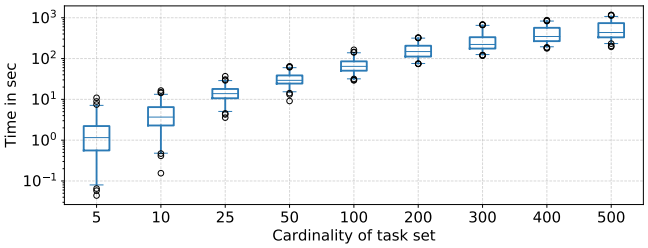


Fig. 7: Runtime of **MC** vs. n. Note the nonlinear x-axis.

E. Scalability to Large Task Sets

In our last experiment, we evaluated the scalability of the **MC** approach with respect to the number of tasks. To this end, we generated 2250 task sets, 50 for each possible combination of $u \in \{0.75, 0.8, 0.85, 0.9, 0.95\}$ and $n \in \{5, 10, 25, 50, 100, 200, 300, 400, 500\}$. For each task set, we measured the runtime of **MC** required to generate 10^5 samples.

The results are shown in Fig. 7. Note that the x-axis of Fig. 7 is nonlinear. For $n \leq 50$, our *sequential* **MC** implementation can sample 10^5 response times in approximately 30 seconds (with a mean of 32 s for $n = 50$ tasks). At the upper end, for $n = 500$, the generation of 10^5 samples took roughly 10 minutes in one case (with a mean of 8.9 minutes for $n = 500$). Looking at median runtimes, it typically took 1.1, 29.0, and 434.2 seconds to generate the required number of samples for task sets with cardinality 5, 50, and 500, respectively. The observed runtimes, while not perfectly linear, clearly show the **MC** approach to scale easily to large task-set sizes, well beyond the reach of any current convolution-based methods.

Finally, it should be noted that Monte Carlo simulation is an *embarrassingly parallel* workload—it is trivial to parallelize the sampling across dozens, hundreds, or even thousands of cores with minimal communication. Our sequential implementation evaluated herein does not yet reflect this potential. As such, it

may be safely assumed that runtime scalability is not a limiting factor of the **MC** approach given modern computing facilities.

VIII. Related Work

Davis and Cucu-Grosjean [17] recently provided a comprehensive survey of probabilistic schedulability analysis techniques. We focus our attention on the most closely related work.

Analyses modeling task execution times as pWCETs trace their roots to work by Tia et al. [55] in 1995, who introduced *Probabilistic Time Demand Analysis* (PTDA) based on the time-demand analysis technique for the deterministic case by Lehoczky et al. [33]. Gardner and Liu [22] extended PTDA to *Stochastic Time-Demand Analysis* (STDA) by accounting for backlog due to jobs that are not finished by their deadline.

STDA relies on the convolution of random variables representing individual jobs. Continuing this line of work, Díaz et al. [20] noticed that exact stochastic analysis of practical real-time systems is infeasible due to excessive computational costs, which lead them to introduce the notion of intentional pessimism captured by Def. 5. This idea was adopted by Refaat and Hladik [50], who introduced resampling techniques that trade some added pessimism for a decrease in the size of the convolved distributions. Following up on this idea, Maxim et al. [43] presented a resampling approach that accrues less pessimism. Specifically, they suggested to keep the largest value as well as the $k-1$ values with the largest probability in the distribution when reducing it to size k, and to reassign the probability of each removed point to the next-largest retained point. While this approach reduces the pessimism compared to Refaat and Hladik's method [50], pessimism still accrues in an uncontrolled manner if the method is applied repeatedly.

Instead of trying to speed up job-level convolution, von der Brüggen et al. [57] proposed to move the convolution operation to the task level, which is possible since all jobs of a task share the same pWCET distribution. Using multinomial distributions, they efficiently calculated each task's total demand distribution (for a given number of jobs in an interval under analysis) and optimized the computation via pruning techniques. While their method thus scales to larger task sets than prior approaches [57], it remains subject to the efficiency and precision issues inherent in convolution. Specifically, von der Brüggen et al. reported runtimes of multiple hours for sets of 25 or more tasks, while simultaneously employing pessimism-inducing state-merging heuristics to lower the memory needs of the method. As the results reported in Section VII strongly suggest that the proposed Monte Carlo approach can match the level of precision attained by task-level convolution for larger task sets (as reported by von der Brüggen et al. [57]) with much lower runtimes, we chose to exclude task-level convolution from consideration in our experimental setup.

Concurrently with this study, Markovic et al. [40] pointed out that implementing *circular convolution* (based on the Fourier transform)—rather than naïve direct convolution as done herein and in most of the just-cited studies, with the notable exception of Gardner and Liu's early work [22]—can considerably lessens the scalability issues of convolution-based approaches. There is thus reason to expect the envelope in which convolution-based methods remain competitive to be pushed further in future work.

Alternatively, analytic approximation techniques provide an immediate upper bound on the probability of a deadline miss without resorting to convolution. Chen and Chen [10] proposed a scalable approximation based on the Chernoff bound [13], which they later further enhanced with regard to both runtime and precision [12]. Furthermore, von der Brüggen et al. [57] presented analyses based on Bernstein's [21] and Hoeffding's [26] inequalities. As discussed in Section VII, it makes sense to try these upper bounds first since they are fast to compute, but they do tend to be quite pessimistic in general.

Lu, Nolte, and several co-authors [35–38] studied the applicability of *statistical methods* to estimate the response time of a job. Their approach runs a system under analysis for some time, records the observed response times, and applies extreme value theory (EVT) to derive a probabilistic bound on the maximum response time. Maxim et al. [44] examined the soundness of such approaches. Note that this approach solves a different problem than this paper: we approximate the *proportion* of response times that exceed the deadline, whereas Lu et al. seek to estimate the worst-case response time.

Finally, *stochastic model checking* (SMC) [53] is a well-established area with a rich literature [1, 31] that may seem conceptually very similar. However, there are major differences in focus and technique. SMC provides an expressive modeling formalism tailored to verifying complex, user-defined properties specified with temporal logics (*e.g.*, [46, 47]). Such flexibility comes with a significant runtime cost, and hence SMC works best for models comprised of a modest number of interacting entities exhibiting complex behavior. In contrast, we have developed—and rigorously justified—a minimal, extremely lightweight sampling procedure tailored specifically to the (much simpler) problem at hand that scales to hundreds of tasks and tens of thousands of jobs. It is unlikely that a general SMC approach could produce samples at a comparably high rate, which is essential to achieving the required high precision δ.

IX. Conclusion and Future Work

We have proposed a novel approach based on Monte Carlo simulation to analyze the probabilistic response time of jobs under preemptive static-priority scheduling on a uniprocessor, and applied it to approximate the WCDFP of constrained-deadline sporadic firm real-time tasks. The essence of the Monte Carlo approach is to accept a minuscule possibility of misestimation for the benefit of a comparably fast runtime. In an evaluation with randomized task sets, the proposed approach was shown to be highly effective compared to the state of the art when given equivalent time budgets, scaling to large task sets with up to 500 tasks while providing better precision than prior analytical and convolution-based approaches.

In future work, it would be interesting to extend the proposed approach to more complex task models (*e.g.*, to self-suspending tasks) and different scheduling policies like EDF, and to examine the applicability to non/limited-preemptive workloads or in situations where pWCETs are correlated.

ACKNOWLEDGEMENTS

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No 803111).

REFERENCES

[1] G. Agha and K. Palmskog, "A survey of statistical model checking," *ACM Trans. Model. Comput. Simul.*, vol. 28, no. 1, pp. 6:1–6:39, 2018.

[2] A. Agresti and B. A. Coull, "Approximate is better than "exact" for interval estimation of binomial proportions," *The American Statistician*, vol. 52, no. 2, pp. 119–126, 1998.

[3] B. Akesson, M. Nasri, G. Nelissen, S. Altmeyer, and R. I. Davis, "A comprehensive survey of industry practice in real-time systems," *Tech. Rep.*, 2020.

[4] N. C. Audsley, A. Burns, M. M. Richardson, K. Tindell, and A. J. Wellings, "Applying new scheduling theory to static priority pre-emptive scheduling," *Softw. Eng. J.*, vol. 8, no. 5, pp. 284–292, 1993.

[5] P. Axer, M. Sebastian, and R. Ernst, "Probabilistic response time bound for CAN messages with arbitrary deadlines," in *Design, Automation & Test in Europe Conference & Exhibition (DATE'12), Dresden, Germany, March 12-16*. IEEE Computer Society, 2012, pp. 1114–1117.

[6] S. Bernstein, "Sur l'extension du théorème limite du calcul des probabilités aux sommes de quantités dépendantes," *Mathematische Annalen*, vol. 97, no. 1, pp. 1–59, 1927.

[7] S. Bozhko and B. B. Brandenburg, "Abstract Response-Time Analysis: A Formal Foundation for the Busy-Window Principle," in *32nd Euromicro Conference on Real-Time Systems (ECRTS'20), July 7-10, 2020, Virtual Conference*.

[8] I. Broster, A. Burns, and G. Rodríguez-Navas, "Probabilistic analysis of CAN with faults," in *Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS'02), Austin, Texas, USA, December 3-5*. IEEE Computer Society, 2002, pp. 269–278.

[9] L. D. Brown, T. T. Cai, and A. DasGupta, "Interval estimation for a binomial proportion," *Statistical science*, vol. 16, no. 2, pp. 101–117, 2001.

[10] K.-H. Chen and J.-J. Chen, "Probabilistic schedulability tests for uniprocessor fixed-priority scheduling under soft errors," in *12th IEEE International Symposium on Industrial Embedded Systems (SIES'17), Toulouse, France, June 14-16*. IEEE Computer Society, 2017, pp. 1–8.

[11] K.-H. Chen, G. von der Brüggen, and J.-J. Chen, "Analysis of deadline miss rates for uniprocessor fixed-priority scheduling," in *24th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'18), Hakodate, Japan, August 28-31,*. IEEE Computer Society, 2018, pp. 168–178.

[12] K.-H. Chen, N. Ueter, G. von der Brüggen, and J.-J. Chen, "Efficient computation of deadline-miss probability and potential pitfalls," in *Design, Automation & Test in Europe Conference & Exhibition (DATE'19), Florence, Italy, March 25-29*. IEEE Computer Society, 2019, pp. 896–901.

[13] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *The Annals of Mathematical Statistics*, vol. 23, no. 4, pp. 493–507, 1952.

[14] C. J. Clopper and E. S. Pearson, "The Use of Confidence or Fiducial Limits Illustrated in The Case of The Binomial," *Biometrika*, vol. 26, no. 4, pp. 404–413, 12 1934.

[15] L. Cucu-Grosjean, "Independence-a misunderstood property of and for probabilistic real-time systems," *In Real-Time Systems: the past, the present and the future*, pp. 29–37, 2013.

[16] R. I. Davis and A. Burns, "Robust priority assignment for messages on controller area network (CAN)," *Real Time Syst.*, vol. 41, no. 2, pp. 152–180, 2009.

[17] R. I. Davis and L. Cucu-Grosjean, "A survey of probabilistic schedulability analysis techniques for real-time systems," *Leibniz Trans. Embed. Syst.*, vol. 6, no. 1, pp. 04:1–04:53, 2019.

[18] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller area network (CAN) schedulability analysis: Refuted, revisited and revised," *Real Time Syst.*, vol. 35, no. 3, pp. 239–272, 2007.

[19] J. L. Díaz, D. F. García, K. Kim, C. Lee, L. L. Bello, J. M. López, S. L. Min, and O. Mirabella, "Stochastic analysis of periodic real-time systems," in *Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS'02), Austin, Texas, USA, December 3-5*. IEEE Computer Society, 2002, pp. 289–300.

[20] J. L. Díaz, J. M. López, M. G. Vazquez, A. M. Campos, K. Kim, and L. L. Bello, "Pessimism in the stochastic analysis of real-time systems: Concept and applications," in *Proceedings of the 25th IEEE Real-Time Systems Symposium (RTSS'04), 5-8 December, Lisbon, Portugal*. IEEE Computer Society, 2004, pp. 197–207.

[21] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*, ser. Applied and Numerical Harmonic Analysis. Birkhäuser, 2013.

[22] M. K. Gardner and J. W. Liu, "Analyzing stochastic fixed-priority real-time systems," in *Tools and Algorithms for Construction and Analysis of Systems, 5th International Conference (TACAS'99), Amsterdam, The Netherlands, March 22-28*. Springer, 1999, pp. 44–58.

[23] D. Griffin, I. Bate, and R. I. Davis, "dgdguk/drs."

[24] ——, "Generating utilization vectors for the systematic evaluation of schedulability tests," in *41st IEEE Real-Time Systems Symposium (RTSS'20), December 1-4, Houston, TX, USA*. IEEE Computer Society, 2020, pp. 76–88.

[25] A. Hamann, D. Dasari, S. Kramer, M. Pressler, and F. Wurst, "Communication centric design in complex automotive embedded systems," in *29th Euromicro Conference on Real-Time Systems (ECRTS'17), June 27-30, Dubrovnik, Croatia*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, pp. 10:1–10:20.

[26] W. Hoeffding, "Probability inequalities for sums of bounded random variables," in *The Collected Works of Wassily Hoeffding*. Springer, 1994, pp. 409–426.

[27] M. Joseph and P. K. Pandya, "Finding response times in a real-time system," *Comput. J.*, vol. 29, no. 5, pp. 390–395, 1986.

[28] C. Kao and H.-C. Tang, "Several extensively tested multiple recursive random number generators," *Computers & Mathematics with Applications*, vol. 36, no. 6, pp. 129–136, 1998.

[29] S. Kramer, D. Ziegenbein, and A. Hamann, "Real world automotive benchmarks for free," in *6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, 2015.

[30] P. L'Ecuyer and F. Panneton, "Fast random number generators based on linear recurrences modulo 2: overview and comparison," in *Proceedings of the 37th Winter Simulation Conference, Orlando, FL, USA, December 4-7*. IEEE Computer Society, 2005, pp. 110–119.

[31] A. Legay, B. Delahaye, and S. Bensalem, "Statistical model checking: An overview," in *Runtime Verification - First International Conference (RV'10), St. Julians, Malta, November 1-4, 2010. Proceedings*. Springer, 2010, pp. 122–135.

[32] J. P. Lehoczky, "Fixed priority scheduling of periodic task sets with arbitrary deadlines," in *Proceedings of the Real-Time Systems Symposium (RTSS'90), Lake Buena Vista, Florida, USA, December*. IEEE Computer Society, 1990, pp. 201–209.

[33] J. P. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: Exact characterization and average case behavior," in *Proceedings of the Real-Time Systems Symposium (RTSS'89), Santa Monica, California, USA, December*. IEEE Computer Society, 1989, pp. 166–171.

[34] J. M. López, J. L. Díaz, J. Entrialgo, and D. F. García, "Stochastic analysis of real-time systems under preemptive priority-driven

scheduling," *Real Time Syst.*, vol. 40, no. 2, pp. 180–207, 2008.
[35] Y. Lu, T. Nolte, J. Kraft, and C. Norström, "A statistical approach to response-time analysis of complex embedded real-time systems," in *16th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'10), Macau, SAR, China, 23-25 August*. IEEE Computer Society, 2010, pp. 153–160.
[36] ——, "Statistical-based response-time analysis of systems with execution dependencies between tasks," in *15th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'10), Oxford, United Kingdom, 22-26 March*. IEEE Computer Society, 2010, pp. 169–179.
[37] Y. Lu, T. Nolte, I. Bate, and L. Cucu-Grosjean, "A statistical response-time analysis of complex real-time embedded systems by using timing traces," in *6th IEEE International Symposium on Industrial Embedded Systems (SIES'11), Vasteras, Sweden, June 15-17*. IEEE Computer Society, 2011, pp. 43–46.
[38] ——, "A statistical response-time analysis of real-time embedded systems," in *Proceedings of the 33rd IEEE Real-Time Systems Symposium (RTSS'12), San Juan, PR, USA, December 4-7*. IEEE Computer Society, 2012, pp. 351–362.
[39] F. Markovic, J. Carlson, R. Dobrin, B. Lisper, and A. Thekkilakattil, "Probabilistic response time analysis for fixed preemption point selection," in *13th IEEE International Symposium on Industrial Embedded Systems (SIES'18), Graz, Austria, June 6-8*. IEEE Computer Society, 2018, pp. 1–10.
[40] F. Markovic, A. V. Papadopoulos, and T. Nolte, "On the convolution efficiency for probabilistic analysis of real-time systems," in *33rd Euromicro Conference on Real-Time Systems (ECRTS'21), July 5-9, 2021, Virtual Conference*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 16:1–16:22.
[41] D. Maxim and L. Cucu-Grosjean, "Response time analysis for fixed-priority tasks with multiple probabilistic parameters," in *Proceedings of the IEEE 34th Real-Time Systems Symposium (RTSS'13), Vancouver, BC, Canada, December 3-6*. IEEE Computer Society, 2013, pp. 224–235.
[42] D. Maxim, L. Santinelli, and L. Cucu-Grosjean, "Improved sampling for statistical timing analysis of real-time systems," *Proceedings of the 18th International Conference on Real-Time Networks and Systems (RTNS'10), Toulouse, France*, 2010.
[43] D. Maxim, M. Houston, L. Santinelli, G. Bernat, R. I. Davis, and L. Cucu-Grosjean, "Re-sampling for statistical timing analysis of real-time systems," in *20th International Conference on Real-Time and Network Systems (RTNS'12), Pont a Mousson, France - November 08 - 09*. ACM, 2012, pp. 111–120.
[44] D. Maxim, F. Soboczenski, I. Bate, and E. Tovar, "Study of the reliability of statistical timing analysis for real-time systems," in *Proceedings of the 23rd International Conference on Real Time Networks and Systems (RTNS'15), Lille, France, November 4-6*. ACM, 2015, pp. 55–64.
[45] D. Maxim, R. I. Davis, L. Cucu-Grosjean, and A. Easwaran, "Probabilistic analysis for mixed criticality systems using fixed priority preemptive scheduling," in *Proceedings of the 25th International Conference on Real-Time Networks and Systems (RTNS'17), Grenoble, France, October 04 - 06, 2017*. ACM, 2017, pp. 237–246.
[46] B. L. Mediouni, "Modeling and Analysis of Stochastic Real-Time Systems," Ph.D. dissertation, Université Grenoble Alpes, 2019.
[47] B. L. Mediouni, A. Nouri, M. Bozga, M. Dellabani, A. Legay, and S. Bensalem, "*S* BIP 2.0: Statistical model checking stochastic real-time systems," in *Automated Technology for Verification and Analysis - 16th International Symposium (ATVA'18), Los Angeles, CA, USA, October 7-10, 2018, Proceedings*. Springer, 2018, pp. 536–542.
[48] N. Navet, Y. Song, and F. Simonot, "Worst-case deadline failure probability in real-time applications distributed over controller area network," *J. Syst. Archit.*, vol. 46, no. 7, pp. 607–617, 2000.
[49] T. Nolte, H. Hansson, and C. Norström, "Probabilistic worst-case response-time analysis for the controller area network," in *Proceedings of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'03), May 27-30, Toronto, Canada*. IEEE Computer Society, 2003, pp. 200–207.
[50] K. S. Refaat and P. Hladik, "Efficient stochastic analysis of real-time systems via random sampling," in *22nd Euromicro Conference on Real-Time Systems (ECRTS'10), July 6-9, Brussels, Belgium*. IEEE Computer Society, 2010, pp. 175–183.
[51] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*, ser. Springer Texts in Statistics. Springer, 2004.
[52] A. Sailer, S. Schmidhuber, M. Deubzer, M. Alfranseder, M. Mucha, and J. Mottok, "Optimizing the task allocation step for multi-core processors within autosar," in *2013 International Conference on Applied Electronics*. IEEE, 2013, pp. 1–6.
[53] K. Sen, M. Viswanathan, and G. Agha, "On statistical model checking of stochastic systems," in *Computer Aided Verification, 17th International Conference (CAV'05), Edinburgh, Scotland, UK, July 6-10, 2005, Proceedings*. Springer, 2005, pp. 266–280.
[54] B. Tanasa, U. D. Bordoloi, P. Eles, and Z. Peng, "Probabilistic response time and joint analysis of periodic tasks," in *27th Euromicro Conference on Real-Time Systems (ECRTS'15), July 8-10, Lund, Sweden*. IEEE Computer Society, 2015, pp. 235–246.
[55] T. Tia, Z. Deng, M. Shankar, M. F. Storch, J. Sun, L. Wu, and J. W. Liu, "Probabilistic performance guarantee for real-time tasks with varying computation times," in *1st IEEE Real-Time Technology and Applications Symposium, Chicago, Illinois, USA, May 15-17, 1995*. IEEE Computer Society, 1995, pp. 164–173.
[56] S. Tobuschat, R. Ernst, A. Hamann, and D. Ziegenbein, "System-level timing feasibility test for cyber-physical automotive systems," in *11th IEEE Symposium on Industrial Embedded Systems (SIES'16), May 23-25, Krakow, Poland*. IEEE Computer Society, 2016, pp. 121–130.
[57] G. von der Brüggen, N. Piatkowski, K.-H. Chen, J.-J. Chen, and K. Morik, "Efficiently approximating the probability of deadline misses in real-time systems," in *30th Euromicro Conference on Real-Time Systems (ECRTS'18), July 3-6, Barcelona, Spain*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, pp. 6:1–6:22.
[58] B. A. Wichmann and I. D. Hill, "Generating good pseudo-random numbers," *Comput. Stat. Data Anal.*, vol. 51, no. 3, pp. 1614–1622, 2006.
[59] E. B. Wilson, "Probable inference, the law of succession, and statistical inference," *Journal of the American Statistical Association*, vol. 22, no. 158, pp. 209–212, 1927.